

# Database and Web Techniques Report



## Web Feed Aggregator

By

Aravinda Baliga B

549886

Web Engineering

Summer Semester – 2019

# Contents

Introduction	-----	3
Application Workflow	-----	5
Technologies Utilized	-----	15
Appendix	-----	19
References	-----	26

# Introduction

In the past, end-users were visiting the provider of their choice each time when they wanted to get their required information such as news, entertainment, blogs, products, and services, etc. This was not an easy method for the users as they were forced to visit the provider every time and there was no guarantee that there will be any new information since their last visit. Later, the providers introduced the Subscription based approach where a user can subscribe to their interested topics using their email-id and every time when there is an update or new information on the particular topic then subscribed users will be notified of the same by sending an e-mail and they can read more about their interested topics or article using the URL link provided in the e-mail, this made things easier for the user as they did not have to visit the provider to get information but the provider themselves will send them the notification on their topic of interest.

In the recent years, It has become very common that a provider provides the information in a certain standard format (most commonly XML format) and that can be used by the consumer for their needs. This functionality has been adopted by many of the third party vendors and they use this information from providers to showcase to the end users. Most of the third parties aggregate the information they obtain from similar providers and display them in a common structured format. Since all the providers use the same standard format it is very easy for the third parties to adopt them. Also, as an end user, it's very easy for us to obtain the information which we need in the common portal from various providers rather than visiting the individual providers to obtain the same information. For example: if we consider the Google News website ([news.google.com](http://news.google.com)), It aggregates the information from the various popular news sources such as CNBC, BBC, Fox CNN, etc. and displays them under common URL and categorizes them into various categories such as World-news, Local-news, Sports, Entertainment, Technology, etc. so it will be very easy for the end user to follow the news of their interest from various sources rather than visiting the different news providers to obtain the same information.

One of the main terminology of the World Wide Web (WWW) that make this possible is known as Web Feed. A web feed is sometimes also called a News Feed. It is an XML formatted file which contains the required information in the various standard tags. Since Web Feed is in XML format, it is Machine Readable and can be processed by any programming language. RSS (**R**eally **S**imple **S**yndication or **R**ich **S**ite **S**ummary) is a popular type of Web Feed and web syndication format. It uses the XML dialect and is a method for delivering the content which changes frequently. It

was first invented by Netscape as they wanted a standard format to share the information. Also, all RSS files will obey to the XML 1.0 standard specification set by W3C (World Wide Web Consortium), it is an organization which sets the standard for the World Wide Web. Many of the news websites, web-blogs, and various other online publisher syndicate their article or content as an RSS feed to the users.

On a high level, RSS document starts with the element `<rss>` it contains the mandatory attribute `Version`, which specifies the version of the RSS to which the document adheres to. The latest version of the RSS is the RSS2.0. The `<rss>` element contains a single `<channel>` element which contains the information about the metadata and its content. Some of the required channel elements are Title (Name of the channel), Link (URL to article or content) and Description (information about the channel) and some of the optional elements are `pubDate`, `category`, `image`, `rating`, etc. Channel may contain a vast number of `<item>` which represents the story. Each item can contain its own element. Some of the elements of the `<item>` are the title, link, author, `pubDate`, source, category, etc.

Another popular web feed format is known as ATOM. ATOM is an XML based Web content and metadata syndication format. The file extension can be `.XML/.ATOM`. ATOM is a relatively new specification but is more robust and has better features than the RSS. ATOM is stricter than RSS and its root tag is called `<feed>`. ATOM was designed to overcome the limitations of RSS and it is steadily gaining the popularity though RSS remains widely used Web Feed standard.

A provider or website who is willing to publish their content using the RSS or ATOM creates a feed file and stores it on their server. These feeds can be created using various available software or manually. A visitor will subscribe to the web feed and the web feed file will be read by the feed reader and displays only the new items from the feed file. There are a lot of advantages to both the Publisher and the Subscribers. For Subscribers, they can get the latest information and without visiting the provider they can get information from different sources in a common place. Also, they can choose to read the stories in which they are interested in. For the Publishers RSS feed includes a link back to their website which will increase in the traffic to their website and easier publishing as it requires minimal information such as title, link, and description. Also, does not require a large database to store the Subscriber-related information.

# Application Workflow

The Web Feed aggregator application combines the latest movie news from various popular sources such as Boxofficemojo, Fandango, Metacritic, etc. and displays the essential information such as Provider, Title, Published Date and Date time on which program detected this information (Fetch Date) in the structured table format. This application has 2 views the first one displays the Web feed data related information as shown in Fig 1.0.

Refresh Web Feed : Delete Old Records between : 06/20/2019 - 06/20/2019

Last Refreshed at : 2019-06-20 11:42:03.710

**Web Feed Movie Data**

PROVIDER	Title	Published Time	Fetch Date
All Providers	The Child's Play Cast Really Loved Messing With Each Other On The Set Of The New Movie	2019-06-20 08:42:48 AM	2019-06-20 11:42:03 AM
Apple Music	Lover Taylor Swift	2019-08-23 12:00:00 AM	2019-06-20 10:00:27 AM
Box	Threads Sheryl Crow	2019-08-30 12:00:00 AM	2019-06-20 10:00:27 AM
Cinema Blend	No 6 Collaborations Project Ed Sheeran	2019-07-12 12:00:00 AM	2019-06-20 10:00:27 AM
Fandango Movies	We Are Not Your Kind Slipknot	2019-08-09 12:00:00 AM	2019-06-20 10:00:27 AM
	FEVER DREAM Of Monsters and Men	2019-07-26 12:00:00 AM	2019-06-20 10:00:27 AM
Apple Music	"Let's Rock" The Black Keys	2019-06-28 12:00:00 AM	2019-06-20 10:00:27 AM
Apple Music	REAL LIFE Emeli Sandé	2019-06-07 12:00:00 AM	2019-06-20 10:00:27 AM
Apple Music	A Bath Full of Ecstasy Hot Chip	2019-06-21 12:00:00 AM	2019-06-20 10:00:27 AM
Apple Music	Futha Heilung	2019-06-28 12:00:00 AM	2019-06-20 10:00:27 AM

Page 1 of 12 10 View 1 - 10 of 119

Set Refresh Time Set Time

Fig 1.0: Web Feed Data

## Table:

Features of the table itself are: It provides some of the basic information such as total entries are present altogether from the various sources, user can control the number of entries that can be displayed in a single page and can modify it as per their choice, total pages of information available in the table. User can navigate to next news page or return back to the previous page easily, can filter as per the need for any of the column such as Title, Provider or Date, filter for any of the specific providers from the provider drop-down list and read only news from that particular provider. If the user requires more information about any of the particular movie news then they can click on the title which has the URL link embedded to it, it will redirect them to the website where it was originally posted and they can read more about that topic over there. By default, the web feed data are sorted based on Fetch Time but the user can click on any of the required column name and sort according to the data of the column.

### Refresh/Update web feed:

As shown in Fig 1.0, we have the refresh button above the table, upon clicking the Refresh button it will fetch the new data from the various RSS and ATOM feed links and dump them into the database. If the published date of the web feed data is older than 30 days from the current day then those data will not be read. Also, if the data is already present in the database then those are skipped in order to avoid duplicate entries. Only fresh web feed data are written into the database. If the data has been successfully written into the database then alert message will be displayed to the user as in Fig 2.0 below. The alert message will also indicate how many new entries were found during the refresh. After the DB has been refreshed, the Table will also get refreshed automatically and display all the new data as well as the old data together. The new data will be displayed at the top of the table.

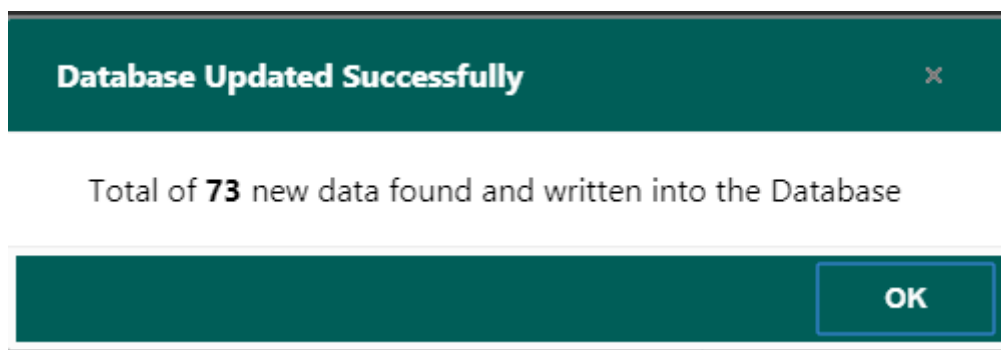


Fig 2.0: Successful Refresh Message

Once the database has been successfully updated then the Refresh button will be inactive for the next 10 minutes so if the user clicks Refresh button again within next 10 mins then the application will display the alert message as shown in Fig 3.0. The alert message will also notify after how many minutes user can refresh the movie news data.

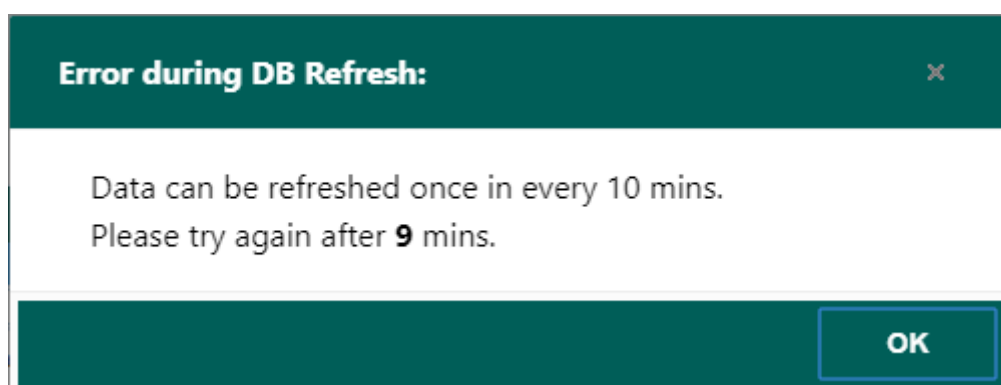


Fig 3.0: Failure of Refresh web feed

### Last Refresh Date-time:

Next to the refresh button user can check the last date and time at which the database was refreshed with new data successfully or date and time of last successful refresh button click as shown in Fig 4.0. Based on this user can decide whether he wants to refresh the data again or not. If the refresh has been done successfully then this date and time will be updated accordingly along with the table so the user can get the clear idea about the refresh time of Web Feed.

Last Refreshed at : **2019-06-10 11:57:19.273**

Fig 4.0: Last Refresh Time

### Delete records based on the given Date:

Along with the Refresh button and last refresh time the user has an option to Delete the unwanted or old movie news data from the database. User has to provide the Start date and End date as shown in Fig 5.0 below, based on this date range web feed data from all providers will be deleted from the database.

at : **2019-06-08 09:40:53.737**

Delete Old Records between : **06/08/2019 - 06/08/2019**

06/08/2019 06/08/2019 Apply Cancel

Jun 2019							Jul 2019						
Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa
26	27	28	29	30	31	1	30	1	2	3	4	5	6
2	3	4	5	6	7	8	7	8	9	10	11	12	13
9	10	11	12	13	14	15	14	15	16	17	18	19	20
16	17	18	19	20	21	22	21	22	23	24	25	26	27
23	24	25	26	27	28	29	28	29	30	31	1	2	3
30	1	2	3	4	5	6	4	5	6	7	8	9	10

Fig 5.0: Delete data date range

As shown in Fig 5.0 user can choose the FROM and TO date, If the TO date is within last 7 days of the current day then deletion is not allowed as the user may try to

delete all the entries from the Database. Hence an error message is shown to the user as in Fig 6.0.

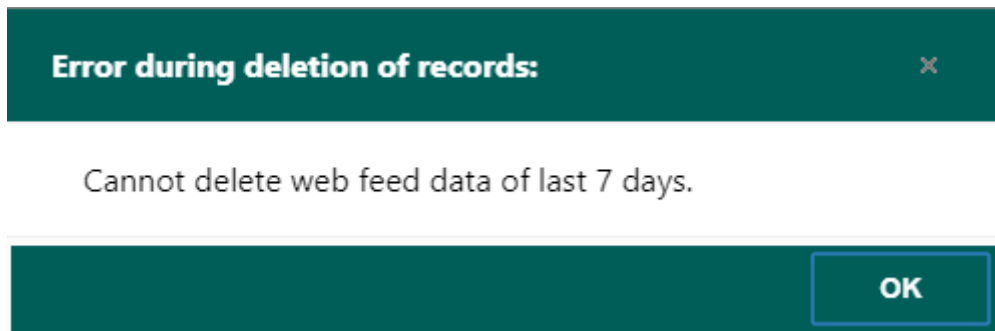


Fig 6.0: Error for Deletion of web feed data

If the TO date does not lie within the last 7 days from the current date, then the web feed data whose Published Date lies between FROM and TO date in the database will be deleted. An alert message is displayed to the user which also indicates total entry that was deleted from the database as shown below in Fig 7.0.

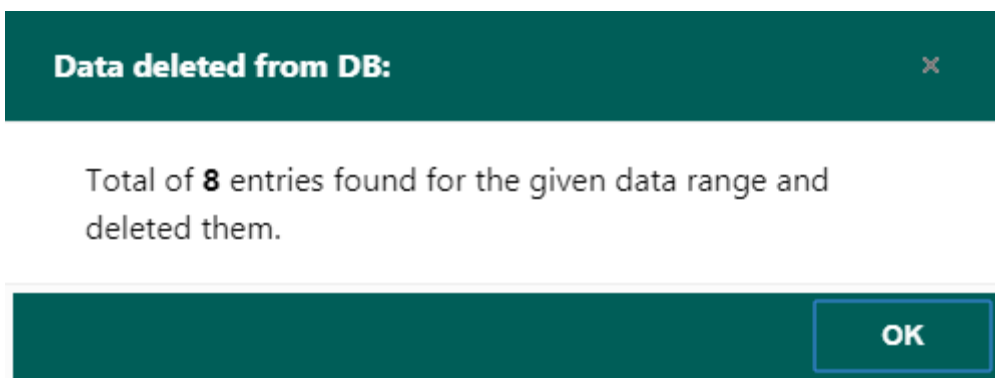


Fig 7.0: Successful deletion message

If no data found for the given date range then following alert message will be displayed and notified the user of the same as shown in below Fig 9.0.

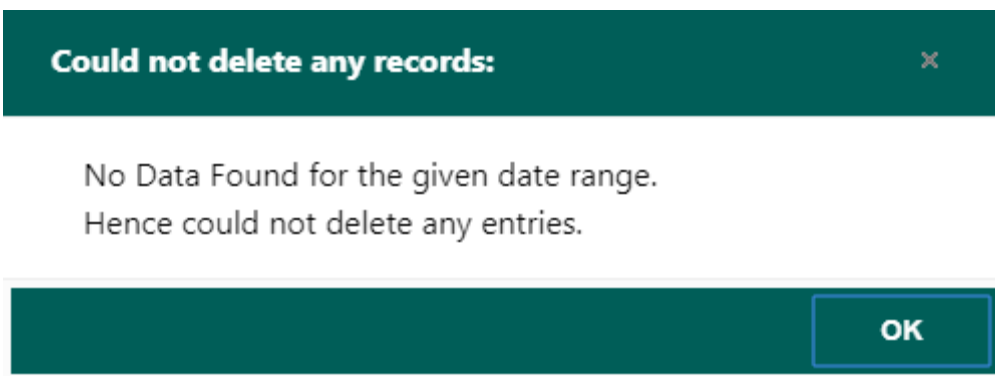


Fig 8.0: No data for deletion message



## Export to Excel:

User has a feature to export the contents of the movie data into Excel by clicking on the Export icon present on the bottom-left side of the table. This will export all the active data present in DB into Excel format which user can utilize for their reference or filter and make modifications as per their need. The exported excel file will have all the important information related to the Movie news Web Feed such as Title, Provider, URL, Published Date, DB Entry date. The exported file will look similar to Fig 9.0.

	A	B	C	D	E	F	G	H	I
1	MOVIE_ID	PROVIDER	TITLE	PUBLISH TIME	DB ENTRY TIME	URL			
2	1	Apple Music	Threads Sheryl Crow	Fri Aug 30 2019 02:00:00	Thu Jun 20 2019 12:00:	https://music.apple.com/ar/album/threads/1467264422?app=music			
3	2	Apple Music	Lover Taylor Swift	Fri Aug 23 2019 02:00:00	Thu Jun 20 2019 12:00:	https://music.apple.com/ar/album/lover/1468058165?app=music			
4	3	Apple Music	We Are Not Your Kind Sli	Fri Aug 09 2019 02:00:00	Thu Jun 20 2019 12:00:	https://music.apple.com/ar/album/we-are-not-your-kind/1463706038?			
5	4	Apple Music	FEVER DREAM Of Monste	Fri Jul 26 2019 02:00:00	Thu Jun 20 2019 12:00:	https://music.apple.com/ar/album/fever-dream/1461520104?app=musi			
6	5	Apple Music	No 6 Collaborations Proje	Fri Jul 12 2019 02:00:00	Thu Jun 20 2019 12:00:	https://music.apple.com/ar/album/no-6-collaborations-project/146454			
7	6	Apple Music	Let's Rock The Black Keys	Fri Jun 28 2019 02:00:00	Thu Jun 20 2019 12:00:	https://music.apple.com/ar/album/lets-rock/1459700588?app=music			
8	7	Apple Music	Futha Heilung	Fri Jun 28 2019 02:00:00	Thu Jun 20 2019 12:00:	https://music.apple.com/ar/album/futha/1457872629?app=music			
9	8	Apple Music	A Bath Full of Ecstasy Hot	Fri Jun 21 2019 02:00:00	Thu Jun 20 2019 12:00:	https://music.apple.com/ar/album/a-bath-full-of-ecstasy/1455652358?			
10	9	MetaCritic Reviews	The Command	Fri Jun 21 2019 00:00:00	Thu Jun 20 2019 11:58:	https://www.metacritic.com/movie/kursk			
11	10	MetaCritic Reviews	Toy Story 4	Fri Jun 21 2019 00:00:00	Thu Jun 20 2019 11:58:	https://www.metacritic.com/movie/toy-story-4			
12	11	MetaCritic Reviews	Toni Morrison: The Pieces	Fri Jun 21 2019 00:00:00	Thu Jun 20 2019 11:58:	https://www.metacritic.com/movie/toni-morrison-the-pieces-i-am			
13	12	MetaCritic Reviews	The Quiet One	Fri Jun 21 2019 00:00:00	Thu Jun 20 2019 11:58:	https://www.metacritic.com/movie/the-quiet-one			
14	13	MetaCritic Reviews	Wild Rose	Fri Jun 21 2019 00:00:00	Thu Jun 20 2019 11:58:	https://www.metacritic.com/movie/wild-rose			
15	14	Cinema Blend	The Child's Play Part 2	Thu Jun 20 2019 10:42:46	Thu Jun 20 2019 12:42:46	https://www.cinemablend.com/news/1475204/the-childs-play-part-2			

Fig 9.0: Export Excel data format

## Export RSS file:

Similar to Excel file, user can export all the Web Feed data present in this application into RSS file using the RSS button present on the bottom-left of the table (Next to Export Excel button). This will convert all the web feed data from various providers present in the database into an RSS feed file of its own as shown in below Fig 10.0.

```
<?xml version="1.0" encoding="utf-8"?>
<rss version="2.0">
  <channel>
    <title>Movie Web Feed Aggregator</title>
    <link>undefined</link>
    <description>Web Feed Data from Various Movie News Sources</description>
    <lastBuildDate>Thu, 20 Jun 2019 21:15:43 GMT</lastBuildDate>
    <docs>http://blogs.law.harvard.edu/tech/rss</docs>
    <generator>https://github.com/jpmonette/feed</generator>
    <item>
      <title><![CDATA[Threads Sheryl Crow]]></title>
      <link>https://music.apple.com/ar/album/threads/1467264422?app=music</link>
      <guid>https://music.apple.com/ar/album/threads/1467264422?app=music</guid>
      <pubDate>Fri, 30 Aug 2019 00:00:00 GMT</pubDate>
      <description><![CDATA[Movie News]]></description>
      <author>BA$VB (BA$VB)</author>
    </item>
  </channel>
</rss>
```

Fig 10.0: Exported RSS File

## Automatic Refresh Data:

User can set the time for an automatic refresh of the web feed data using the text field provided below the table as shown in Fig 11.0.

Fig 11.0: Automatic Refresh time input

While setting the time, the user has to set the time of more than 5 minutes, this has been implemented to avoid frequent refresh of the data because frequent refresh may reduce the performance of the application. Also, the user cannot set the automatic refresh time of more than 60 minutes. If the time is less than 5 minutes or more than 60 minutes then an error message is shown to the user as in Fig 12.0 below.

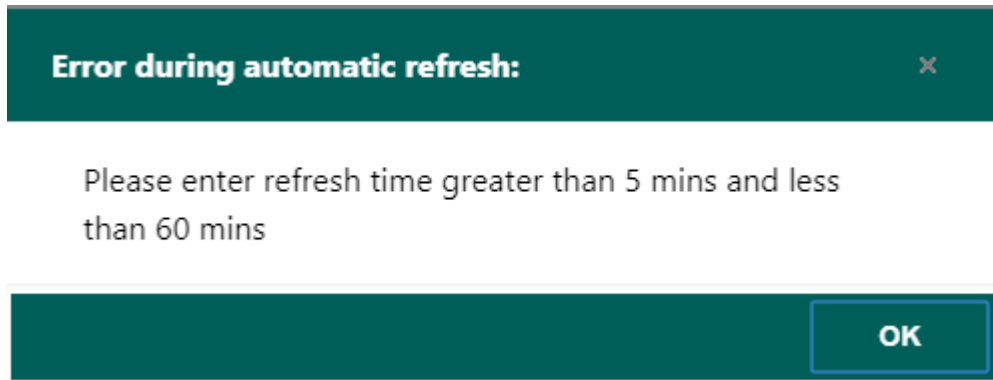


Fig 12.0: Error during automatic refresh time setting

If the time is more than 5 minutes and less than 60 minutes, then a success message will be displayed to the user as shown in Fig 13.0. After every "X" minutes set by the user, data from all the web feed URL will read. If any new data is found which are not older than 30 days then they will be written into the database and table is refreshed automatically with new data.

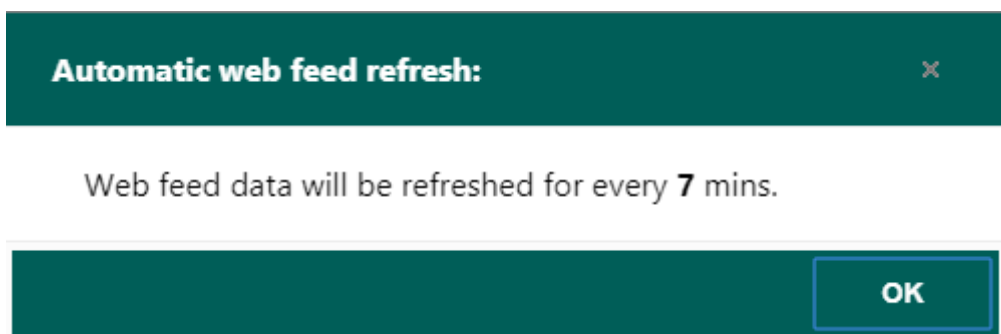


Fig 13.0: Automatic refresh confirmation

### Providers View:

As mentioned earlier there are 2 views to this application. The first view provides the web feed data information and the second view displays the information related to the Provider. Below Fig 14.0 shows the Providers view which gives the information such as Provider Name, URL, Count of new items found during the last refresh for

each provider and date time of the last refresh for each provider. The user can also Add, Modify or Delete the information related to the provider here.

Provider Name	Provider URL	Newly Found	Last Refresh	Edit
Blend	<a href="https://www.cinemablend.com/rss/topic/news/movies">https://www.cinemablend.com/rss/topic/news/movies</a>	0	2019-06-18 11:57:45 AM	
Apple Music	<a href="https://rss.itunes.apple.com/api/v1/in/music-videos/top-mus">https://rss.itunes.apple.com/api/v1/in/music-videos/top-mus</a>	0	2019-06-18 11:57:45 AM	
Movie Web	<a href="https://movieweb.com/rss/movie-news/1">https://movieweb.com/rss/movie-news/1</a>	0	2019-06-18 11:57:45 AM	
Modern Warfare	<a href="https://themovievault.net/rss/channel/modern-warfare-3">https://themovievault.net/rss/channel/modern-warfare-3</a>	0	2019-06-18 11:57:45 AM	
Fandango	<a href="https://www.fandango.com/rss/newmovies.rss">https://www.fandango.com/rss/newmovies.rss</a>	0	2019-06-18 11:57:45 AM	

Fig 14.0: Provider Information

#### Addition of new Web Feed URL:

If the user is interested in receiving information from the new web feed URL (provider) which is in either RSS or ATOM format then they can add them to the application. User can provide the Name and URL of the provider in the fields provided above the table as shown in Fig 15.0.

Fig 15.0: New Provider Information

After clicking the Add URL button, First Provider Name and Provider URL is verified to confirm it is not empty, If it's empty then error message will be displayed to the user. If it is not empty then the Provider URL is read and verified if this is a valid URL with valid RSS/ATOM data using the rss2json API, If the provided URL is not valid then an Error message is displayed to the user as shown in Fig 16.0 indicating he is trying to fetch the data from an invalid Web Feed.

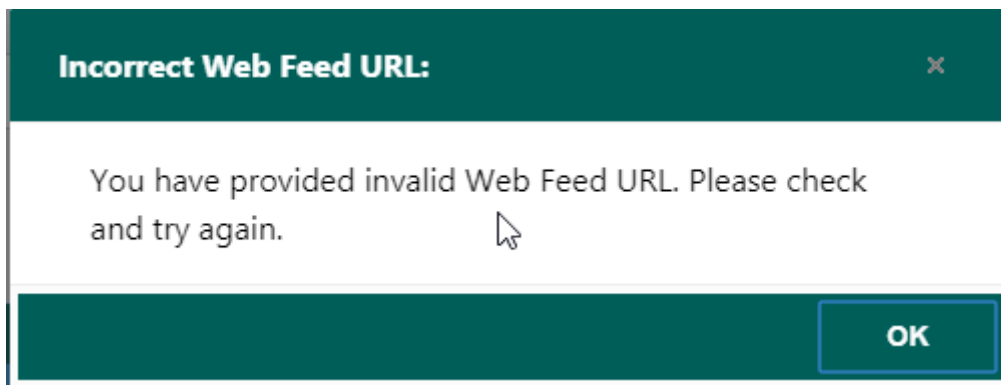


Fig 16.0: Error during the addition of new URL

If the provided URL is valid, then the provider name and provider URL is matched with existing provider names and provider URLs. If they match with any of the existing data then again error message is displayed to the user as shown in Fig 17.0.

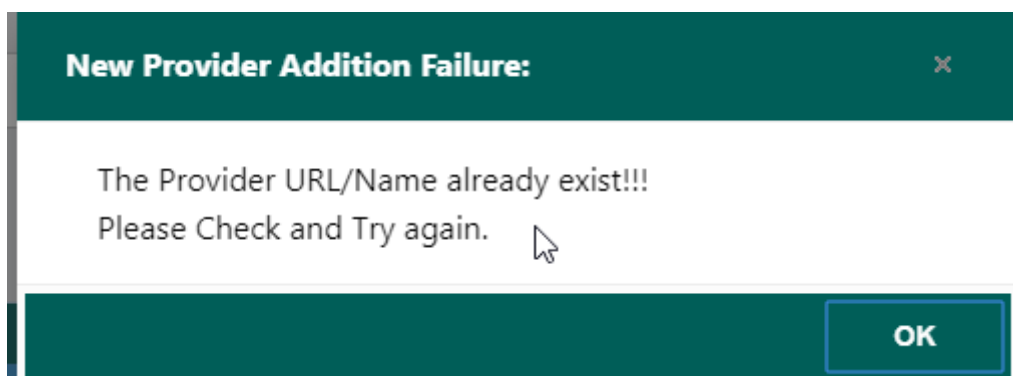


Fig 17.0: Error during the addition of new URL

If the Provider name and URL is unique then the Provider information is added to the database and Provider table is updated with this new Provider information. A confirmation message is displayed to the user as shown in Fig 18.0.

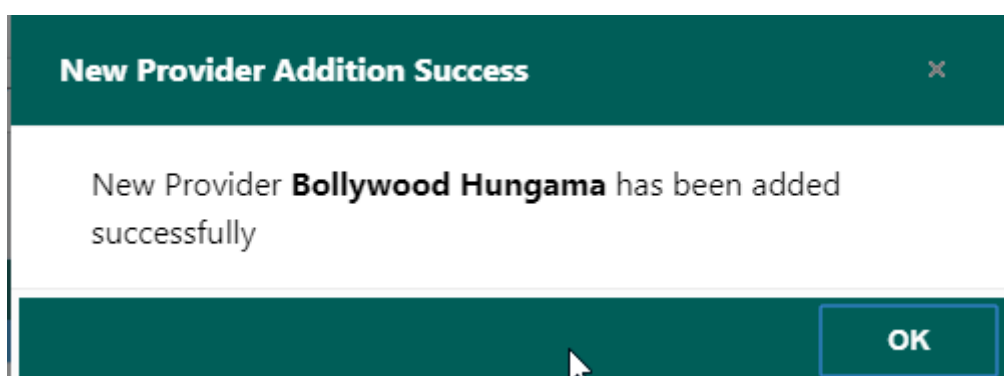


Fig 18.0: Successful addition of new provider

After the successful addition of the new Provider, the web feed data from all the providers including the new provider will be read and if any new entries are found then that new information will be written into the database. Also, the Web Feed

Data (Fig 1.0) table is updated to reflect the new web feed data from new provider along with the old data.

### Modify Provider Information:

User can modify the information such as Provider Name and Provider URL using the modify button present against each of the Provider. Upon clicking the “Edit Provider” blue button, a modal will be displayed with the current Provider name and URL as shown in Fig 19.0.

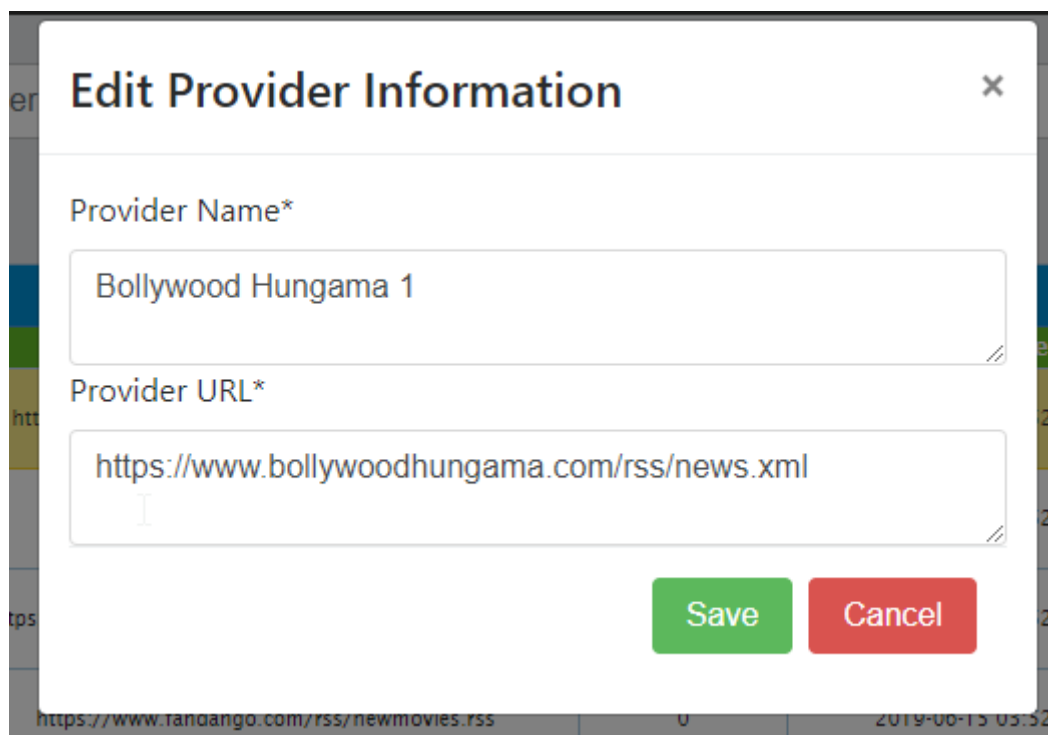
A screenshot of a web application modal titled "Edit Provider Information". The modal has a close button (X) in the top right corner. It contains two text input fields. The first field is labeled "Provider Name\*" and contains the text "Bollywood Hungama 1". The second field is labeled "Provider URL\*" and contains the text "https://www.bollywoodhungama.com/rss/news.xml". At the bottom right of the modal, there are two buttons: a green "Save" button and a red "Cancel" button. The modal is overlaid on a background that shows a table with columns for provider information and a timestamp "2019-06-15 05:52".

Fig 19.0: Edit Provider Information modal

User can modify the information and submit them using the Save button. If the URL of the Provider or Name matches with the existing Provider Name or URL then error message will be displayed to the user. If the modified information is unique then URL is verified to check if it's valid using the API rss2json. If it is not valid then an error message will be displayed to the user otherwise the information is saved. The Provider table, as well as the Web Feed data table, is refreshed with the newly provided data to reflect the new information.

### Delete Provider Information:

User has the feature to delete the provider completely if he is not interested in receiving the web feed data from any of the Provider. User can click on the “Delete Provider” button present against each of the provider. Upon clicking the delete button, a popup will be displayed as shown in Fig 20.0 to confirm the selection, if the user confirms by clicking on the Ok button then the particular provider is deleted permanently. Also, the web feed data which were present from this provider is deleted from the database. The Provider table and the Web Feed data table is refreshed with the new information.

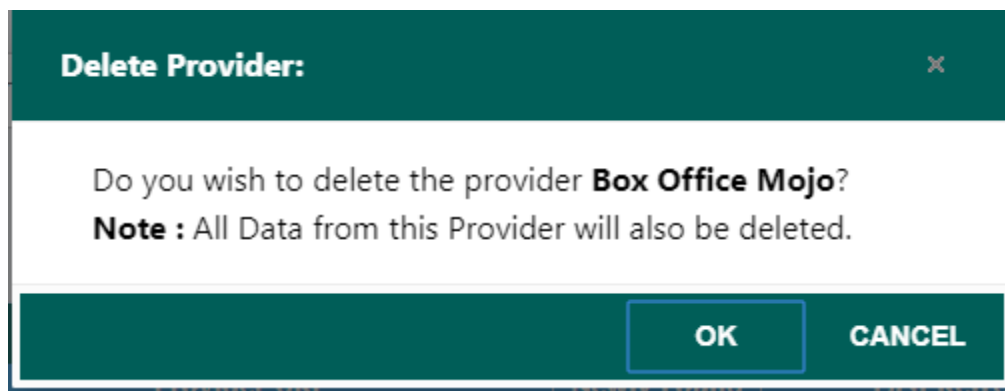


Fig 20.0: Delete Provider

If the user wants to add the same Provider again then he can use the Add Provider feature as explained earlier and start receiving the web feed data from the same provider again.

### Delete records older than 30 days:

The SQL server database deletes the web feed entries whose published date is older than 30 days from the current day. This has been implemented using the Windows task scheduler. At the beginning of every day (around 00:30) a SQL Server Stored Procedure “DELETE\_OLD\_RECORDS” is executed automatically using the Windows task scheduler.

```
sqlcmd -S BVBALIGA\SQLEXPRESS -U sa -P myPass -d Data_And_Web_SS19 -Q " EXEC DELETE_OLD_RECORDS "
echo Data Older Than 30 are deleted from the table
pause
```

As shown above, these commands are present in the .bat file which will trigger the Stored procedure and delete the old records. This .bat file has been included in the Windows Task Scheduler and it will run every day so the data older than 30 days will be removed automatically.

# Technologies Utilized

During the development of Web application choice of technologies is very important because the Web application should be durable for a long time and if the technologies do not support the new features and functionality then it would be very difficult to adopt some of the requirement into our application. Also, sometimes it is very difficult to choose from a vast amount of available technologies. Hence it is very important to understand the requirement correctly before making the final decision. Following table 1.0 shows the various technologies utilized in the Web Feed Movie news aggregator Web application.

Database	Backend
MS SQL Server 2012	Node.js 10.12.0
Rest Interface	Frontend
GET, POST, DELETE	HTML 5.0, CSS 3.0, AJAX, JQuery 2.2.4, Bootstrap, Jqgrid 4.14.1, Font-awesome, Alertify.

Table 1.0: Technologies used

## MS SQL Server:

Microsoft Structure Query Language is one of the most popular relational database management system (RDBMS). It was released in 1989 as version 1.0. Most of the industries use MS SQL Server as a database for their application because it's stable, easy to use and it provides a lot of inbuilt functionality which makes the usage very easy. I have used it as my database in this application mainly because all the RSS web feed will have the same data structure hence using the relational database would be a better option compared to NOSQL. Also, I have used a lot of functionalities such as Stored Procedure, Index, Triggers, etc. which makes the execution easy and very fast when we have a huge amount of data such as the web feed data from various sources.

## Node.js:

As we know Javascript has been one of the most popular programming languages for client-side application development. The extension of the javascript to the server-

side execution is done by Node.js. It is asynchronous and it operates on a single threaded event. Node.js has been booming in recent years and having knowledge would be really helpful in terms of career growth so I choose the Node.js. The other reasons for using Node.js in this application is its rich NPM(Node Package Manager) library which provides a huge number of packages, these are very easy to install and use. Also, it uses the javascript syntax which makes implementation easy. As mentioned before it's Asynchronous hence it does not wait for the task or operation to complete so it will be very fast compared to other languages such as PHP. Along with Node.js, I have used Express.js framework which allows the middleware setups and routing is defined using this framework. So Express.js and Node.js provide the strong backend functionality which is very important for any Web application which involves frequent database usage such as this Web Feed Movie News aggregator application.

### **Frontend:**

As we know frontend is the user's view of the Web application, It is the part normally user gets to see and interact. It mainly consists of HTML, CSS, and Javascript. HTML (Hyper Text Markup Language) is the backbone of the web and every site we visit has been built using the HTML. HTML has gone through profound changes over the years and currently, we use HTML5. In this application, I have used the HTML to display the table, button, calendar and various other elements which we see in the home page of the application. CSS (Cascading Style Sheets) is the controller for the looks of the HTML page. It defines the styling attributes for elements such as button, text, images, etc. We can also use CSS to arrange the elements of the HTML page however we want. In this web application, I have used the CSS sheet for the coloring of the table, button, navbar, and alignment of various other elements we see in the application. It is normally recommended to use external style sheet rather than using the inline style or using style elements within the header of HTML page hence I have used the external style sheet which has been included in the HTML head part. Javascript is a scripting language which is normally used to create a dynamic and interactive web application. The programs in javascript language are called scripts. They can be written right in a web page's HTML and gets loaded automatically as the page loads. In this application, I have used one of the popular javascript extension known as jQuery and I have made separate jQuery file and that has been included in the HTML page. jQuery is a javascript library which makes it easier for the developers to use javascript in their application. jQuery compresses the code into a single



function. I have used jQuery for all the interactions such as Refresh Data, Insert Data, Delete Data, etc. Along with the basic necessary things such as HTML, CSS, and jQuery I have also used Bootstrap which is the famous framework for CSS. It helps in making the application responsive. The latest version of Bootstrap is Bootstrap 4. It uses the concept of a Grid to make the application responsive. I have used Bootstrap mainly to make the application responsiveness so that it will appear the same in different kind of platforms such as Computer, Mobiles, tablets, etc. Along with these technologies, one last concept which makes the frontend complete is known as AJAX (Asynchronous Javascript and XML). It is not a programming language rather a technique which can be used to send or receive data from the Database. It acts as a link between frontend and backend program. It allows the exchange of data in the background and primary advantage is that it restricts the page reload. Using AJAX we can reload the part of the web page as per our need. Due to this AJAX functionality, web page performance and usability increase drastically. In this application, AJAX has been used to load the Data from the Database into the table, Refresh the table and date of the last update. If you notice this application, during none of the operations the entire page is reloaded, rather only the specific part of the web page such as a table is reloaded. Usage of AJAX is very easy with the jQuery. So altogether these techniques make up for the frontend of my Movie News Web Feed aggregator, These technologies increase the performance and overall appearance of the application. Also, most importantly they make it easier for the user to interact with the application and get information from it in an easy way.

#### **jqGrid:**

jqGrid provides a solution for representing and manipulating tabular data on the web. It is an AJAX-enabled jQuery plugin. It is a client-side representation and loading of the data is done through AJAX callback. It can be used with any of the server side technology such as PHP, Perl, Node.js, etc. Since it is a Javascript library it is supported by most of the browsers. In this application, we are representing the structure data which has limited columns and data hence I have used the jqGrid, it contains four essential columns such as Provider, Title, Published Date and Fetched date for the web feed data. It has a lot of inbuilt functionality which makes it easier for the developer as well as the user.

#### **Font-Awesome:**

Font-Awesome Icon and Font toolkit. It is built on top of CSS and LESS. is an is a plugin which can be included in the application. It provides various Icons and symbols which can be used with inline elements to increase the appearance of the

application. As we know Icons and Symbols are more appealing to the user compared to the text. Hence the inclusion of font-awesome reduces the text and beautifies the application. I have used font-awesome to indicate some of the icons such as Refresh button, Modify button and Delete button.

### Alertify:

Alert messages are a very important part of the web application. It provides various information to the users such as the success of a particular event or error occurred during the execution of an event. The normal alert message is displayed using the browser's default functionality which is not very much appealing to the user hence I have used the Alertify package. It provides an alert message to the user in an advanced way by overwriting the default functionality. The main objective of using the Alertify is to increase the appearance of the application. During the explanation of the application workflow, we got to know that there is a lot of Alert message which will be displayed to the user hence it is important to make sure they are appealing to the user. Please refer below Fig 21.0 and 22.0, which provides a comparison between default alert message and Alertify alert message.

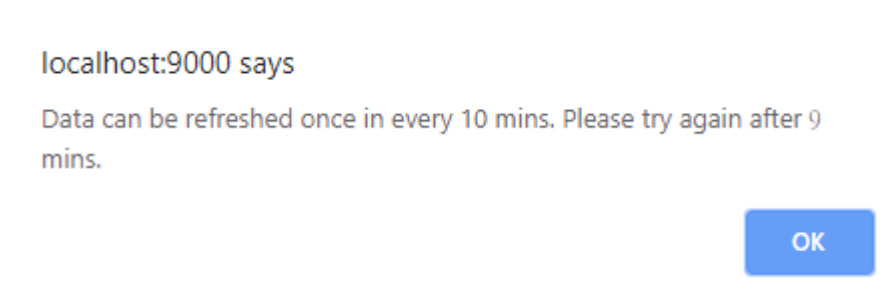


Fig 21.0: Default alert message dialog

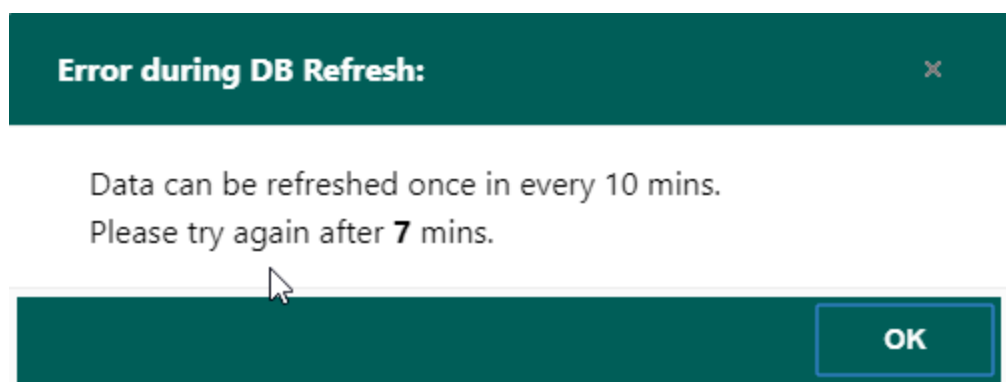


Fig 22.0: Alertify alert message dialog

# Appendix

## **Rest Interface:**

REST (Representational State Transfer) or RESTful API is built to take the benefits of some of the existing protocols. It primarily takes advantage of HTTP (Hypertext Transfer Protocol). It uses the common HTTP requests such as GET (retrieve the data from Server), POST (add new data to Server), DELETE (remove data from Server), PUT (Update existing data in Server). REST was introduced in 2000 and it is flexible. Unlike SOAP which is bound to XML format, in REST we can use different formats such as XML, JSON, YAML, etc. The methods GET, PUT and DELETE are Idempotent which means that making the same request to server more than once does not make multiple changes to the server rather only once the changes will be affected. The POST method is not idempotent as making the same POST request will end up in making changes to the server every time. Hence making the same POST request to the server will result in the creation of new entries every time. REST is stateless protocol and it uses standard operations which will increase the performance, reliability, and reduces redundant code by re-using the components. REST API in this application acts as a link between the database and the front-end, it will fetch the data from a database or send the data to the database. In this application, I have used the GET, POST, and DELETE for our operations and in all the cases I am expecting JSON data as return data from the Server.

## **GET:**

GET method is used to retrieve resource or information. GET method do not make any changes to the Server in any way. GET method returns with a code 200(OK) response if the data is found else with the code 404 (Not Found).

## **DELETE:**

As the method name suggests it is used to remove the entries from the server. In order to remove the entries, we need to pass some parameters based on which the data can be removed from our database. If the operation is a success then it returns the response code of 200(OK).

## **POST:**

The POST method is used to create new entries on the server side. Normally POST method is used to send the data of the form elements or uploading the file contents. GET method receives the data from server whereas POST request sends data to the server.

Following are the REST API requests which I have made use in this application, All these REST APIs make use of callback function as Node.js is asynchronous.

### Home Page render:

As shown in Appendix 1.0, the GET method is used to display the Index page of the application to the user. When the user enters the URL in the browser by default this GET method is triggered for the initial load. This application is running on the port 9000 so when a user enters localhost:9000 then the index page is loaded first using this REST API. This method does not take any parameter, it just returns the index page in HTML format which has been placed in the public folder of the application.

```
//Display the Index page when user hits the URL in browser
app.get('/', function(req,res){
    res.render('index.html');
});
```

Appendix 1.0: Display Home Page

### Fetch new data:

As shown below in Appendix 2.0, the POST method will refresh the table with new data from the various Web Feed URL. This method will trigger the function dumpdata which will read all the Web Feed URL first and then feed data from these URLs. If there are any new data found, then those data will be written into the database and jqGrid table is refreshed to update the new entries. This POST method takes the Refresh\_Time parameter which will verify if the time is greater than 10 minutes. This function will return the data in JSON format to the front-end which can be fed to the jqGrid table.

```
//Refresh the Database with fresh data onclick of the REFRESH button
app.post('/Refresh_Data', function(req,res){
    var Time_Interval = req.body.Refresh_Time;
    webFeedDump.dumpdata(Time_Interval,function(data){
        res.end(data);
    });
});
```

Appendix 2.0: Get web feed data

### Load data in the table:

As shown in Appendix 3.0, GET method will retrieve all the Web Feed data information from the database and provide the data to the jqGrid table in JSON format which will display the information in the specified structured format. This

method is triggered as soon as the user navigates to the URL localhost:9000 in the browser. This GET method does not take any parameter, it will fetch all the data present in the database. Since the Node.js is asynchronous it makes use of the callback method meaning that after the completion of the function execution Web\_Feed\_Data present in the controller webFeedRetrieve, the data will be returned to the calling function.

```
//Display the data in JQGRID on page load
app.get('/Web_Feed_Data', function(req,res){
    webFeedRetrieve.Web_Feed_Data(function(data){
        res.send(data);
    });
});
```

Appendix 3.0: Refresh web feed data

### Export data:

The GET method in Appendix 4.0 is triggered using the URL “/Export\_Movie\_Data” to Export all the Web Feed data into the Excel. This GET method does not take any parameter and upon clicking the export button, this GET method is triggered. This method will just make a call to the function “Export\_Movie\_Data” which will read the data from the database and create an Excel report with headers, this will be automatically downloaded through the user's browser.

```
//Onclick of the Export button Export the Data into Excel
app.get('/Export_Movie_Data', function(req,res){
    webFeedExport.Export_Movie_Data(req,res);
});
```

Appendix 4.0: Export web feed data to Excel

### Delete Records:

The REST API DELETE method is used to remove the entries from Database based on the date range provided by the user. As shown in below Appendix 5.0, It accepts the Start date and End date as the two parameters which will be passed to the function deletedata present in the controller webFeedDelete and the entries between these dates will be deleted from the database. After deleting the records using stored procedure, it will return success indicator in JSON format and this will be returned to the calling AJAX function using the function res.send() method in Node.js.

```
//Delete the Data from DB as per the Date provided by the user
app.delete('/Delete_Data', function(req,res){
    var Start_Date = req.body.Start_Date;
    var End_Date = req.body.End_Date;
    webFeedDelete.deletedata(Start_Date,End_Date,function(data){
        res.send(data);
    });
});
```

Appendix 5.0: Delete Web Feed data

### Display Refresh Time:

To display the last refresh time to the user I am making use of the GET method. This method does not take any parameter and it will execute the function Last\_Refresh\_Time present in the controller webFeedDump, which will query the database and fetch the last refresh time and return it to the front end in JSON format using the res.send() function. Using the jQuery this JSON data is parsed and displayed to the user.

```
//Get the Last refresh time of the Database
app.get('/Last_Refresh_Time',function(req,res){
    webFeedDump.Last_Refresh_Time(function(data){
        res.send(data);
    });
});
```

Appendix 6.0: Display last refresh date time

### Automatic Refresh of Web Feed:

This POST method shown in Appendix 7.0 will refresh the Web Feed table content automatically based on the Time interval provided by the user. This method will take 2 parameters one is the Time interval provided by the user and another parameter is the End date (which was used for the deletion of web feed data and it can be empty) so if the User\_Delete\_End\_Date is populated then Web Feed data prior to that date will not be read and inserted into the database. After the insertion of all new data into the database, a JSON formatted data is returned to the front end and the table is reloaded again to reflect these new data.

```
//Refresh the Data of the table based on time interver provided by user
app.post('/Refresh_Data_Time_Interval',function(req,res){
    var Time_Interval = req.body.Refresh_Time;
    var User_Delete_End_Date = req.body.User_Delete_End_Date;
    webFeedDump.dumpdata(Time_Interval,User_Delete_End_Date,function(data){
        res.send(data);
    });
});
```

Appendix 7.0: Refresh web feed data automatically

### Load Providers data in the table:

The GET method using the URL /Provider\_Info is used to retrieve all the Provider information such as Provider name, Provider URL, Newly found data count and last refresh of the URL. This method does not take any parameters, This will just query the database for Providers information and it will return all the information to the front end in JSON format. After receiving the data, jqGrid will display the data in a structured format. This works similar to the Web Feed data retrieval.

```
//Get the Provider Information for the Providers JQGRID
app.get("/Provider_Info",function(req,res){
    webFeedProviders.Provider_Table_Load(function(data){
        res.send(data);
    });
});
```

Appendix 8.0: Display web feed Provider's information

### Addition of New Provider:

The POST method shown in Appendix 9.0 is used to insert a new Provider to the database. This POST method takes two parameters from the calling function Provider\_Name and Provider\_URL both the information will be provided by the user. This information is then sent to the database query and inserted into the database. After the insertion, a success indicator will be sent using the res.send() function to indicate the provider has been added successfully and the table is reloaded again to reflect the new Provider information. If the data is already present then an error response is sent to the front-end.

```
//Add the items data (Title, Provider, URL) of new URL to Database
app.post('/ADD_NEW_URL', function(req,res){
    var Provider_Name = req.body.Provider_Name;
    var Provider_URL = req.body.Provider_URL;
    webFeedProviders.New_Provider_Info(Provider_Name,Provider_URL,function(data){
        res.send(data);
    });
});
```

Appendix 9.0: Add new web feed Provider

### Fetch New Provider Web Feed data:

After the new Provider has been successfully added, the web feed data from the new provider is automatically read by the program without any trigger from the user and it will be written into the database using the POST method shown in the Appendix 10.0. This method will take the input parameter User\_Delete\_End\_Date which will have the End date (if the user has deleted any of the previous records using the delete feature in the application, this can be empty) and based on this, data are

written into the database. After the insertion of Web Feed data, the table is refreshed again to reflect the new Providers web feed data along with the other provider's data. The success indicator is sent as a return to the calling AJAX function in JSON format to provide the confirmation on the same to the user.

```
//After the Addition of the New Provider Read the Data and Insert into the DB
app.post("/New_Provider_Data_Insert",function(req,res){
    var User_Delete_End_Date = req.body.User_Delete_End_Date;
    webFeedDump.New_Provider_Data_Insert(User_Delete_End_Date,function(data){
        res.send(data);
    });
});
```

Appendix 10.0: Load new provider web feed data

### **Fetch Provider Information for Modification:**

When a user clicks on the modification button against a particular provider then the existing information related to the provider such as Provider name and Provider URL should be displayed to the user in the modal. In order to fetch this information, I am making use of the POST method as shown in Appendix 11.0. This method will take the unique Provider\_ID as a parameter using which the provider information is fetched from the database and returned to the front end in JSON format. This information is parsed and displayed to the user in the modal.

```
//Fetch the Provider Details for the Modifications
app.post("/Fetch_Provider_Details",function(req,res){
    var Provider_Id = req.body.Provider_Id;
    webFeedProviders.Provider_Details_Fetch(Provider_Id,function(data){
        res.send(data);
    });
});
```

Appendix 11.0: Fetch provider information

### **Update modified details of Provider:**

After the user modifies Provider information, this updated information should be written into the database. In order to achieve this, I am using the POST method as demonstrated in Appendix 12.0. This method takes req.body as an input parameter which will have all the modified information such as Provider Name, Provider URL, Provider\_ID. If the modified data matches with existing provider information then an error is returned in JSON format, else the information is updated. After updating, it will return the success message in JSON format using the Node.js inbuilt function res.send() to the front-end which will be displayed to the user to indicate the successful modification.



```
//Insert the Edited Information of the Provider
app.post("/Provider_Edit_Insert",function(req,res){
    webFeedProviders.Provider_Edit_Insert(req.body,function(data){
        res.send(data);
    });
});
```

Appendix 12.0: Update provider information

### Delete Provider Information:

User can remove the Provider and it's related web feed data from the application if he is not interested in receiving the web feed from any of the providers. To remove the Provider, I am making use of the REST API POST method as shown in Appendix 13.0. Whenever the user clicks on the Delete button of particular Provider, it will trigger this method and this method takes the unique Provider ID as the input parameter. Later this function will trigger Delete\_Provider function present in the controller webFeedProviders and it will pass the Provider\_Id to the function. The respective provider information will be deleted from the database and web feed data related to that provider is no longer available to the user. The success indicator is sent to the front end in JSON format, to provide the confirmation of provider deletion to the user.

```
//Delete the Provider From the table and its related Information
app.post("/Delete_Provider",function(req,res){
    var Provider_Id = req.body.Provider_Id;
    webFeedProviders.Delete_Provider(Provider_Id,function(data){
        res.send(data);
    });
});
```

Appendix 13.0: Delete provider

### Export to RSS:

Below shown Appendix 14.0 is used to export the web feed data into RSS file. This functionality makes use of the GET method and it does not take any parameter and returns a file with RSS data. This GET method will trigger a function and it will first create a file and query the database, returned Web feed data are then written into the file in RSS format. Later this file will be automatically downloaded through the Users browser.

```
//On click of the Export RSS button Export the Data into RSS
app.get('/Export_RSS', function(req,res){
    webFeedExport.Export_Movie_Data_RSS(req,res);
});
```

Appendix 14.0: Export to RSS

# References

## ❖ Introduction to Web Feed and RSS:

1. <https://www.tutorialspoint.com/rss/what-is-rss.htm>
2. <https://perishablepress.com/an-easy-introduction-to-web-feeds/>

## ❖ How does an RSS file looks like and meaning of its element:

1. [https://www.w3schools.com/xml/xml\\_rss.asp](https://www.w3schools.com/xml/xml_rss.asp)
2. <https://www.xul.fr/en-xml-rss.html>

## ❖ How does an RSS works:

<https://computer.howstuffworks.com/internet/basics/rss.htm>

## ❖ How to find RSS and its internal workflow:

1. <https://rss.softwaregarden.com/aboutrss.html>
2. <https://www.lifewire.com/what-is-an-rss-feed-4684568>

## ❖ ATOM Web Feed introduction:

1. <https://www.tutorialspoint.com/rss/what-is-atom.htm>
2. [https://www.ibm.com/support/knowledgecenter/en/SSGMCP\\_5.2.0/com.ibm.ci.cs.ts.internet.doc/topics/dfhtl\\_atom\\_what.html](https://www.ibm.com/support/knowledgecenter/en/SSGMCP_5.2.0/com.ibm.ci.cs.ts.internet.doc/topics/dfhtl_atom_what.html)

## ❖ Advantages of Web Feed:

<https://www.tutorialspoint.com/rss/rss-advantages.htm>

## ❖ Code related issues and verification:

1. <https://stackoverflow.com/>
2. <https://api.rss2json.com/v1/api.json>